Probabilistic Guidance of Swarms using Sequential Convex Programming*

Daniel Morgan[1], Giri Prashanth Subramanian[1], Saptarshi Bandyopadhyay[1], Soon-Jo Chung[1] and Fred Y. Hadaegh[2]

***Abstract*— In this paper, we integrate, implement, and validate formation flying algorithms for large number of agents using probabilistic swarm guidance with inhomogeneous Markov chains and model predictive control with sequential convex programming. Using an inhomogeneous Markov chain, each agent determines its target position during each time step in a statistically independent manner while the swarm converges to the desired formation. Moreover, the swarm is robust to external disturbances or damages to the formation. An optimal control problem is formulated to ensure that the agents reach the target positions while avoiding collisions. This problem is solved using sequential convex programming to determine optimal, collision-free trajectories and model predictive control is implemented to update these trajectories as new state information becomes available. Finally, we validate the probabilistic swarm guidance and model predictive control algorithms using the formation flying testbed.**

I. Introduction

There has been a lot of research interest in the guidance, navigation and control of formation flying agents. [1]–[4]. Using formation flying, multiple small satellites can out perform a single monolithic satellite in certain applications like interferometry. It is now technically feasible to launch and deploy 1000s of small 100-gram satellites, called femtosats, into Earth orbit so that they can reconfigure into desired formations [5]. Similarly, swarms of robots are suitable for applications like environmental monitoring [6], reconnaissance of dangerous or unknown regions while overcoming obstacles [7], and collectively building, sorting or foraging objects of interest [8]. In this paper, we integrate and implement distributed guidance, navigation and control (GNC) algorithms for such large swarms of agents and validate them using our formation flying testbed (FFT).

Instead of the traditional view of guidance of multi-agent systems which deals with an indexed collection of agents, we adopt an *Eulerian* view in this paper as we control the swarm density distribution of a large number of index-free agents over disjoint bins [9]–[14]. Reconfiguration of such large swarms is achieved using probabilistic swarm guidance using inhomogeneous Markov chains (PSG) [15]. In probabilistic swarm guidance, each autonomous agent or robot determines its target position during each time step in a statistically independent manner using a Markov chain [16]–[18]. The swarm converges to the desired formation after multiple time steps and is robust to external disturbances or damages to the formation. The key concept is to design an inhomogeneous Markov chain with the desired formation as its stationary distribution, where the Markov matrices tend to an identity matrix to ensure that the agents settle down after the desired formation has been achieved [15], [19].

The main contribution of this paper is the integration of PSG with a path-planning algorithm. Model predictive control using sequential convex programming (MPC–SCP) is used to control the agents, so that they reach the target positions, assigned by PSG, while avoiding collisions. Recently, convex optimization [20] has been used in multi-vehicle trajectory design and shown that it can be efficiently solved to achieve a global optimum by state-of-the-art interior point methods. Convex optimization has been used to implement a receding horizon controller for a convex problem [21]. Additionally, convex optimization has been used to find collision-free trajectories for a formation reconfiguration [22] and robotic motion planning [23]. However, convexifying the collision constraints results in an overly conservative approximation of the collision avoidance region. Sequential convex programming has also been used for collision avoidance of a quadcopter fleet [24]. In this paper, sequential convex programming (SCP) [25] is implemented using model predictive control (MPC) to provide real-time, collision-free trajectory generation for agents in the swarm in the presence of moving or undetected obstacles. MPC–SCP is executed multiple times to ensure that convex approximations of non-convex constraints are accurately captured, resulting in optimal trajectories [26]. The PSG–MPC and MPC–SCP algorithms are discussed in detail in Section II and III.

II. Probabilistic Swarm Guidance

We adopt an *Eulerian* approach, as we control the swarm density distribution over the state space. PSG involves designing an inhomogeneous Markov chain so that each autonomous agent determines its own trajectory in a statistically independent manner, while the swarm converges to the desired formation and is robust to external disturbances.

Let the convex, compact state space $\mathcal{X} \subset \mathbb{R}^{n_x}$ be partitioned into N_{cell} convex, disjoint bins $(\mathcal{R}\{p\}, p =$

*This research was supported in part by AFOSR grant FA95501210193 and a NASA Space Technology Research Fellowship (NNX11AM84H). This research was carried out in part at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

[1]D. Morgan, G. P. Subramanian, S. Bandyopadhyay and S.-J. Chung are with the Department of Aerospace Engineering, University of Illinois at Urbana-Champaign (UIUC). Email: {morgan29, gpsubra2, bandyop2, sjchung}@illinois.edu.

[2]F. Y. Hadaegh is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA. Email: fred.y.hadaegh@jpl.nasa.gov

$1, \ldots, N_{\text{cell}}$) so that $\bigcup_{p=1}^{N_{\text{cell}}} \mathcal{R}\{p\} = \mathcal{X}$. Let $N \in \mathbb{N}$ agents belong to this swarm and the row vector $\mathbf{r}_\ell^j$ represent the bin in which the j^{th} agent is actually present at the ℓ^{th} PSG iteration. If $\mathbf{r}_\ell^j\{p\} = 1$, then the j^{th} agent is inside $\mathcal{R}\{p\}$ at the ℓ^{th} time instant; otherwise $\mathbf{r}_\ell^j\{p\} = 0$. The ensemble mean of actual agent positions gives the current swarm distribution, i.e., $\mathcal{F}_\ell^\star := \frac{1}{N}\sum_{j=1}^N \mathbf{r}_\ell^j$. The desired formation shape is represented by a probability (row) vector $\boldsymbol{\pi} \in \mathbb{R}^{N_{\text{cell}}}$ over the bins in $\mathcal{X}$. Note that $\boldsymbol{\pi}$ can be any arbitrary probability vector, but it is the same for all agents within the swarm. Note that the agents can estimate the current swarm distribution ($\mathcal{F}_\ell^j$) in a distributed manner by communicating with neighboring agents and using the consensus algorithm [27], where $\|\mathcal{F}_\ell^j - \mathcal{F}_\ell^\star\| \leq \epsilon_{\text{consensus}}$ and $\epsilon_{\text{consensus}}$ is the desired consensus error.

The Hellinger distance (HD) is a symmetric measure of the difference between two probability distributions and it is upper bounded by 1 [28]. The tuning parameter (ξ_ℓ^j) is the HD between the estimate of the current swarm distribution ($\mathcal{F}_\ell^j$) and the desired formation ($\boldsymbol{\pi}$), using the following equation [15]:

$$\xi_\ell^j = D_H(\boldsymbol{\pi}, \mathcal{F}_\ell^j) := \frac{1}{\sqrt{2}}\sqrt{\sum_{p=1}^{N_{\text{cell}}}\left(\sqrt{\boldsymbol{\pi}\{p\}} - \sqrt{\mathcal{F}_\ell^j\{p\}}\right)^2}. \tag{1}$$

Let $\mathbf{d}_\ell^j \in \mathbb{R}^{N_{\text{cell}}}$ denote the predicted position of the j^{th} agent at the ℓ^{th} PSG iteration, where $\mathbf{d}_\ell^j\{p\} = \mathbb{P}(\mathbf{r}_\ell^j\{p\} = 1), \forall p = 1, \ldots, N_{\text{cell}}$. The elements of the row stochastic Markov transition matrix $M_\ell^j \in \mathbb{R}^{N_{\text{cell}} \times N_{\text{cell}}}$ are the transition probabilities of the j^{th} agent at the ℓ^{th} PSG iteration [15]:

$$M_\ell^j\{p, q\} := \mathbb{P}\left(\mathbf{r}_{\ell+1}^j\{q\} = 1 | \mathbf{r}_\ell^j\{p\} = 1\right). \tag{2}$$

The Markov transition matrix M_ℓ^j determines the time evolution of the pmf row vector $\mathbf{d}_\ell^j$ as $\mathbf{d}_{\ell+1}^j = \mathbf{d}_\ell^j M_\ell^j, \forall \ell \in \mathbb{Z}^*$. Let $\boldsymbol{\alpha}_\ell^j \in \mathbb{R}^{N_{\text{cell}}}$ be a nonnegative bounded column vector with $\|\boldsymbol{\alpha}_\ell^j\|_\infty \leq 1$. For given ξ_ℓ^j from (1), the following parametrized family of row stochastic Markov matrices M_ℓ^j have $\boldsymbol{\pi}$ as their stationary distribution (i.e., $\boldsymbol{\pi} M_\ell^j = \boldsymbol{\pi}$) [15]:

$$M_\ell^j = \text{diag}(\boldsymbol{\alpha}_\ell^j)\mathbf{1}\frac{\xi_\ell^j}{\boldsymbol{\pi}\boldsymbol{\alpha}_\ell^j}\boldsymbol{\pi}\text{diag}(\boldsymbol{\alpha}_\ell^j) + \mathbf{I} - \xi_\ell^j\text{diag}(\boldsymbol{\alpha}_\ell^j), \tag{3}$$

where $\boldsymbol{\pi}\boldsymbol{\alpha}_\ell^j \neq 0$ and $\sup_\ell \xi_\ell^j\|\boldsymbol{\alpha}_\ell^j\|_\infty \leq 1$. The $\boldsymbol{\alpha}_\ell^j$ vector can incorporate physical distance between bins in the following manner $\forall p = 1, \ldots, N_{\text{cell}}$ [15]:

$$\boldsymbol{\alpha}_\ell^j\{q\} := 1 - \frac{dis(\mathcal{R}\{q\}, \mathcal{R}\{p\})}{\max_{s=1,\ldots,N_{\text{cell}}} dis(\mathcal{R}\{s\}, \mathcal{R}\{p\}) + 1}, \tag{4}$$

where $dis(\mathcal{R}\{q\}, \mathcal{R}\{p\})$ is the distance between the bins $\mathcal{R}\{q\}$ and $\mathcal{R}\{p\}$, and p satisfies $\mathbf{r}_\ell^j\{p\} = 1$.

If each agent executes the PSG–MPC algorithm illustrated in Method 1, then each $\mathbf{d}_\ell^j$ asymptotically converges pointwise to $\boldsymbol{\pi}$ [15]. The swarm distribution $\mathcal{F}_\ell^\star$ also asymptotically converges pointwise to $\boldsymbol{\pi}$, due to the strong law of large numbers (c.f. [29, pp. 85]). If $\lim_{\ell\to\infty} \mathcal{F}_\ell^\star = \boldsymbol{\pi}$, then $\lim_{\ell\to\infty} \xi_\ell^j = 0$ and $\lim_{\ell\to\infty} M_\ell^j = \mathbf{I}$. Note that the $\boldsymbol{\alpha}_\ell^j$ vector can also incorporate motion constraints [15]. The convergence and stability guarantees of the proposed PSG–MPC algorithm are provided in [15].

Method 1 [15] Probabilistic swarm guidance algorithm using model predictive control (PSG–MPC)

$\ell = 0$
while Swarm is operational **do**
  for all j **do**
    Agent determines its present bin, e.g., $\mathbf{r}_\ell^j\{p\} = 1$
    Agent estimates the current swarm distribution $\mathcal{F}_\ell^j$
    Compute the tuning parameter ξ_ℓ^j using (1)
    Compute the $\boldsymbol{\alpha}_\ell^j$ vector using (4)
    Compute the Markov matrix M_ℓ^j using (3)
    Generate random number $z \in \text{unif}[0; 1]$
    $s^j = s$ satisfying $\sum_{q=1}^{s-1} M_\ell^j\{p, q\} \leq z < \sum_{q=1}^{s} M_\ell^j\{p, q\}$
  end for
  Run Method 2 with terminal bins $\mathcal{R}\{s^j\}$
  Update the desired swarm distribution $\boldsymbol{\pi}$
  $\ell = \ell + 1$
end while

III. Model Predictive Control with Sequential Convex Programming

The main contribution of this paper is using MPC–SCP to provide collision-free trajectories for each PSG iteration. Generating the collision-free trajectories is an important aspect of the guidance and control algorithm. In addition to transferring the agents to their desired terminal positions, the guidance algorithm must also avoid collisions between agents. In order to achieve these goals, model predictive control using sequential convex programming [26] (MPC–SCP) is used.

The objective of the trajectories used in the FFT is to minimize the acceleration of each agent while satisfying the initial and terminal points, maintaining a safe distance between each pair of agents, and using feasible velocities and accelerations. Therefore, we can define the problem as follows.

Problem 1 (Nonlinear Optimal Control Problem):

$$\min_{\mathbf{u}^j, j=1,\ldots,N} \sum_{j=1}^{N} \int_0^{t_f} \|\mathbf{u}^j(t)\|_2 \, dt \quad \text{subject to} \tag{5}$$

$$\ddot{\mathbf{x}}^j(t) = \mathbf{u}^j(t), \quad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{6}$$

$$\|\mathbf{u}^j(t)\|_2 \leq U_{\max}, \quad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{7}$$

$$\|D\mathbf{x}^j(t)\|_2 \leq V_{\max}, \quad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{8}$$

$$\|C[\mathbf{x}^j(t) - \mathbf{x}^i(t)]\|_2 \geq R_{\text{col}} \tag{9}$$
$$\forall t \in [0, t_f], \qquad i > j, \qquad j = 1, \ldots, N-1$$

$$\mathbf{x}^j(0) = \mathbf{x}_0^j, \qquad \mathbf{x}^j(t_f) \in \mathcal{R}\{s^j\} \qquad j = 1, \ldots, N \tag{10}$$

$$\mathbf{x}^j(t) \in \mathcal{X}, \quad \forall t \in [0, t_f], \qquad j = 1, \ldots, N \tag{11}$$

where $U_{\max}$ is the maximum allowable acceleration, $V_{\max}$ is the maximum allowable velocity, R_{col} is the collision radius, $C = [\mathbf{I}_{3\times3} \quad \mathbf{0}_{3\times3}]$, and $D = [\mathbf{0}_{3\times3} \quad \mathbf{I}_{3\times3}]$.

In order to efficiently solve for the trajectories of the agents for the FFT, Problem 1 must be converted to a convex program. It is important to note that the objective function (5) and the constraints of (6), (7), (8), (10), and (11) already satisfy the requirements for a convex programming problem. Therefore, only the collision avoidance constraints, (9), need to be converted in order to make Problem 1 convex. The details for convexifying this problem can be found in [26]. The main steps are discretizing the problem using a zero-order hold and linearizing the collision avoidance constraint described in (9).

The convex programming representation of Problem 1 is written as follows (for spacecraft j).

Problem 2 (Convex Problem):

$$\min_{\mathbf{u}^j} \sum_{k=0}^{T-1} \|\mathbf{u}^j[k]\|_2 \Delta t \quad \text{subject to} \tag{12}$$

$$\mathbf{x}^j[k+1] = A\mathbf{x}^j[k] + B\mathbf{u}^j[k], \quad k = 0,\dots,T-1 \tag{13}$$
$$\|\mathbf{u}^j[k]\|_2 \le U_{\max}, \qquad k = 0,\dots,T-1 \tag{14}$$
$$\|D\mathbf{x}^j[k]\|_2 \le V_{\max}, \qquad k = 0,\dots,T-1 \tag{15}$$
$$(\bar{\mathbf{x}}^j[k] - \bar{\mathbf{x}}^i[k])^T C^T C(\mathbf{x}^j[k] - \bar{\mathbf{x}}^i[k]) \ge R_{\text{col}}\|C(\bar{\mathbf{x}}^j[k] - \bar{\mathbf{x}}^i[k])\|_2 \tag{16}$$
$$k = 0,\dots,T, \qquad i \in \mathcal{I}^j$$
$$\mathbf{x}^j[0] = \mathbf{x}_0^j, \qquad \mathbf{x}^j[T] \in \mathcal{R}\{s^j\} \tag{17}$$
$$\mathbf{x}^j[k] \in \mathcal{X}, \qquad k = 0,\dots,T \tag{18}$$

where $\mathbf{x}^j[k] = \mathbf{x}^j(t_k)$, $\mathbf{u}^j[k] = \mathbf{u}^j(t_k)$, $\bar{\mathbf{x}}^j[k]$ is the nominal trajectory about which the collision avoidance constraint is linearized, and

$$A = \begin{bmatrix} I_{3\times3} & \Delta t I_{3\times3} \\ \mathbf{0}_{3\times3} & I_{3\times3} \end{bmatrix} \tag{19}$$
$$B = \begin{bmatrix} \Delta t I_{3\times3} & \frac{1}{2}\Delta t^2 I_{3\times3} \end{bmatrix} \tag{20}$$
$$\mathcal{I}^j = \{i | i < j\}$$

Now that the optimal control problem has been written as a convex program, SCP (lines 3–23 of Method 2) can be used to efficiently solve the nonlinear optimal control problem in Problem 1. In order to develop a real-time, trajectory-generating algorithm that can account for sensor and actuator errors as well as avoid moving or newly discovered obstacles, MPC-SCP is applied. To describe the MPC–SCP algorithm, a new optimization, Problem 3, is defined. Problem 3 is defined so that there is a finite horizon (T_H). Collision avoidance for Problem 3 is only considered before the end of the horizon. In Problem 3, the spacecraft are assumed to have limited communication ranges. Therefore, they can only communicate with their neighboring spacecraft. Problem 3 is expressed as follows.

Problem 3 (Convex Optimization used in MPC–SCP):

$$\min_{\mathbf{u}^j} \sum_{k=k_0}^{k_0+T_H-1} \|\mathbf{u}^j[k]\|_2 \Delta t_1 + \sum_{k=k_0+T_H}^{T-1} \|\mathbf{u}^j[k]\|_2 \Delta t_2 \tag{21}$$

subject to

$$\mathbf{x}^j[k+1] = A\mathbf{x}^j[k] + B\mathbf{u}^j[k], \quad k = k_0,\dots,T-1 \tag{22}$$
$$\|\mathbf{u}^j[k]\|_2 \le U_{\max}, \qquad k = k_0,\dots,T-1 \tag{23}$$
$$\|D\mathbf{x}^j[k]\|_2 \le V_{\max}, \qquad k = k_0,\dots,T-1 \tag{24}$$
$$(\bar{\mathbf{x}}^j[k] - \bar{\mathbf{x}}^i[k])^T C^T C(\mathbf{x}^j[k] - \bar{\mathbf{x}}^i[k]) \ge R_{\text{col}}\|C(\bar{\mathbf{x}}^j[k] - \bar{\mathbf{x}}^i[k])\|_2, \tag{25}$$
$$k = k_0,\dots,k_0+T_H, \quad \{i,j\} : i \in \mathcal{N}^j,$$
$$\mathcal{N}^j = \{i | i < j, \|\mathbf{x}^j[k_0] - \mathbf{x}^i[k_0]\|_2 \le R_{\text{comm}}\}$$
$$\mathbf{x}^j[k_0] = \mathbf{x}^j_{\text{actual}}, \qquad \mathbf{x}^j[T] \in \mathcal{R}\{s^j\} \tag{26}$$
$$\mathbf{x}^j[k] \in \mathcal{X}, \qquad k = k_0,\dots,T \tag{27}$$

where Δt_1 and Δt_2 are the time step size before and after the end of the horizon, respectively, and R_{comm} is the communication or sensing radius.

The MPC–SCP algorithm reduces the horizon of the convex programs and then solves this problem repeatedly throughout the reconfiguration. Initially, SCP is run to determine optimal trajectory up to a finite horizon (T_H). As the spacecraft nears this horizon in real time, a new trajectory is calculated starting from the current time (k_0) and position ($\mathbf{x}^j_{\text{actual}}$) until the new horizon ($k_0 + T_H$). It is important to note that k_0 is the current time at the beginning of each MPC iteration and varies with time. $\mathbf{x}^j_{\text{actual}}$, in (26), is the real-time state of the spacecraft when SCP begins. This process is repeated until the spacecraft reaches the desired bin ($\mathcal{R}\{p^j\}$) at the final time (T). This process is shown in more detail in Method 2.

The result of the MPC–SCP algorithm is a fully decentralized optimal guidance algorithm with better robustness to sensor and actuator errors. The decentralization of the trajectory generation greatly reduces the communication and computation requirements.

IV. Simulation Results

In this section, simulation results are presented for a swarm reconfiguration. In this simulation, the PSG–MPC algorithm (Method 1) is used to reconfigure a random swarm of 200 agents into an "I"-shaped formation. The simulation uses a 5×5 grid with 2500 mm bin sides and is run for 35 iterations. The trajectory generation for each iteration, using MPC–SCP (Method 2), uses 20 time steps of 0.5 s and a collision radius of 250 mm. The results of this simulation are shown in Fig. 1–Fig. 3.

Fig. 1–Fig. 2 show how the swarm distribution changes as the number of PSG iterations increases. In Fig. 1, the top plot shows that the HD (1) decreases as the number of PSG iterations increases. This result shows that the swarm distribution is converging to the desired distribution. The bottom plot shows that the number transitions also decreases with increasing PSG iterations. This result demonstrates that the agents move less frequently as the swarm converges to the desired distribution. Additionally, Fig. 2 shows the swarm distribution at several PSG iterations. As the PSG iteration (ℓ) increases, the swarm distribution converges to the desired

Method 2 MPC-SCP [26]

$k_0 = 0$
while $k_0 \leq T$ **do**
  $\bar{\mathbf{x}}^j[k] := \mathbf{0}_{6\times 1},\ \forall j, k$
  $\mathcal{I}^j := \emptyset,\ \forall j$
  $\mathbf{x}_0^j[k]$:= the solution to Problem 3 excluding (25), $\forall j, k$
  $\bar{\mathbf{x}}^j[k] := \mathbf{x}_0^j[k],\ \forall j, k$
  Update $\mathcal{N}^j$ using (25), $\forall j$
  $\mathcal{K} := \{1, \ldots, N\}$
  $m := 1$
  while $\mathcal{K} \neq \emptyset$ **do**
    for all $j \in \mathcal{K}$ **do**
      $(\mathbf{x}_m^j[k], \mathbf{u}_m^j[k])$:=the solution to Problem 3, $\forall k$
    end for
    for all j **do**
      Update $\mathcal{N}^j$ using (25)
      $\bar{\mathbf{x}}^j[k] := \mathbf{x}_m^j[k],\ \forall k$
      if $\|\mathbf{x}_m^j[k] - \mathbf{x}_{m-1}^j[k]\|_\infty < \epsilon\ \forall k$ and $\|C(\mathbf{x}_m^j[k] - \mathbf{x}_m^i[k])\|_2 \geq R_{\text{col}}\ \forall k, \forall i \neq j$ **then**
        Remove j from $\mathcal{K}$
      end if
    end for
    $m := m + 1$
  end while
  $M := m - 1$
  Apply $(\mathbf{x}_M^j[k], \mathbf{u}_M^j[k]),\ \forall j, k = k_0, \ldots, T_H$ until a new trajectory is generated.
  Update k_0 to current time
end while

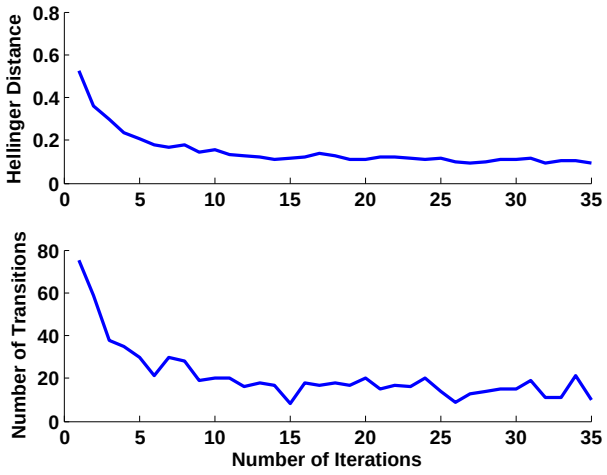

Fig. 1. Hellinger distance after every PSG iteration (top) and number of transitions occurring during each PSG iteration (bottom).

"I" shape. Initially, the agents move into the bins that form the "I" and as the number of iterations increases, the agents become more evenly distributed within the "I".

Fig. 3 shows the number of agents that come within a distance of 250 mm (blue circle), 252.5 mm (red square),

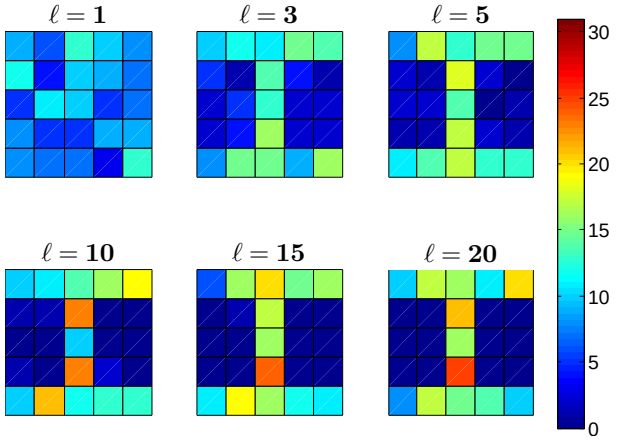

Fig. 2. Number of agents in each bin at PSG iterations 1, 3, 5, 10, 15, and 20. The color of each bin corresponds to the number of agents in that bin.

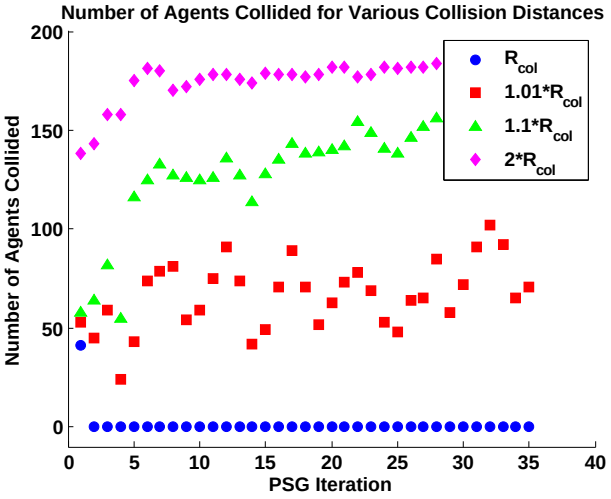

Fig. 3. Number of agents coming within a certain distance of another agent during each PSG iteration.

275 mm (green triangle), and 500 mm (magenta diamond) of another agent during each PSG iteration. At 250 mm, which is the collision radius, there are no collisions after the first PSG iteration. Therefore, MPC–SCP is generating collision-free trajectories for all of the agents at every PSG iteration. The nonzero value during the first PSG iteration is due to the fact that the swarm is randomly initialized so some of the agents violate the collision constraint at the first time step. As can be seen in the 252.5 mm, many of the agents come very near collision radius of another agent, which means that many of the collision constraitns in the MPC–SCP algorithm (Method 2) are active.

V. Experimental Results

In this section, we also demonstrate that these GNC algorithms can be run in real time by implementing them on our FFT. Several testbeds already exist for validating formation flying of a few agents (ten or less agents) [30]– [32]. The

FFT experimental setup is composed of the VICON motion capture system, the communication boards for controlling the agents and the hardware for implementing the GNC algorithms.

Experimental results are presented for three scenarios. The first demonstration uses SCP as an offline, path planning algorithm for several agents in an environment containing fixed obstacles. SCP calculates trajectories that take the agents from their starting positions to their target positions while avoiding each other and the fixed obstacles. In the second demonstration, the MPC–SCP algorithm (Method 2) is used to generate collision-free trajectories in real time in the presence of other agents and fixed obstacles. Additionally, the agents have a limited sensing radius for detecting obstacles and a limited communication radius for detecting other agents. In the third demonstration, the PSG–MPC algorithm determines the target positions for the agents during each time step and the MPC–SCP algorithm provides the real-time, collision-free trajectories to move the agents to the desired positions.

A. Path Planning using SCP

The SCP path planning demonstration uses four agents and four fixed obstacles. The collision radius between agents is 350 mm and the collision radius between an agent and an obstacle is 500 mm. In this demonstration, the fixed obstacles (vertical wooden bars) are set up in a square with 1600 mm sides and the helicopters are located outside of the square near the corners. The target position for each agent is located outside of the opposite corner of the square (markers on the floor). This layout requires each agent to cross diagonally through the square simultaneously while avoiding collisions with the other agents and the obstacles. This layout is shown in Fig. 4.

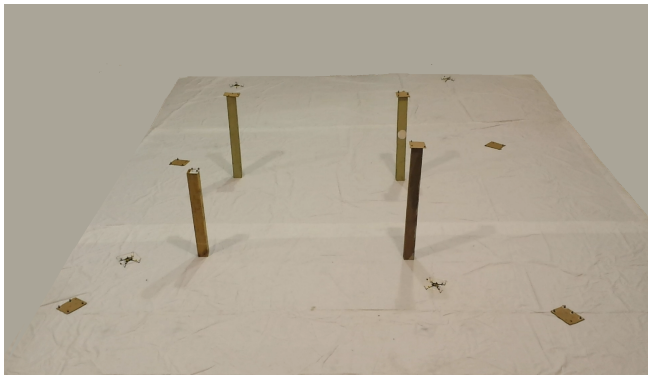

Fig. 4. Quadrotor configuration for SCP demonstration.

Fig. 5 shows the reference trajectories (dashed lines) produced by SCP and the actual trajectories (circles) traversed by the agents in relation to the obstacles. The circle around the obstacle represents its collision radius. The reference trajectories calculated by SCP maintain the required distance from the obstacles while taking each agent to its target position. Additionally, the agents follow the reference trajectories accurately with the largest error being less than 200 mm.

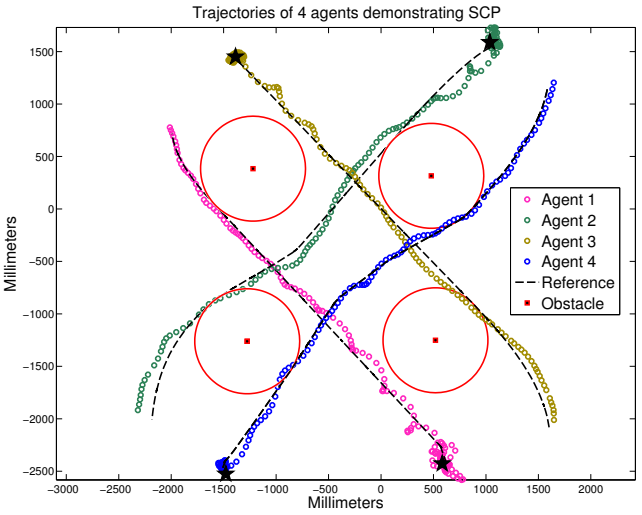

Fig. 5. Path planning reference and actual trajectories for the SCP demonstration. Note that ★ represents the beginning of each trajectory.

B. Real-Time Collision Avoidance using MPC–SCP

The MPC–SCP path planning demonstration uses three agents and one fixed obstacles. The collision radii are the same as they were in the SCP demonstration: 350 mm between agents and 500 mm between an agent and an obstacle. In this demonstration, the fixed obstacle (vertical wooden bars) is located in the center and the helicopters are located in a triangular shape around the obstacle. The target position (markers on the floor) for each agent is located on the opposite side of the obstacle. This layout requires each agent to move around the fixed obstacle while also avoiding the other agents. This layout is shown in Fig. 6.

Fig. 6. Quadrotor configuration for MPC–SCP demonstration.

Fig. 7 shows the reference trajectories (dashed lines) produced by the MPC–SCP algorithm (Method 2) and the actual trajectories (circles) traversed by the agents in relation to the obstacle. The circle around the obstacle represents its collision radius. The reference trajectories calculated by MPC–SCP maintain the required distance from the obstacles while taking each agent to its target position. Additionally, the agents follow the reference trajectories accurately with

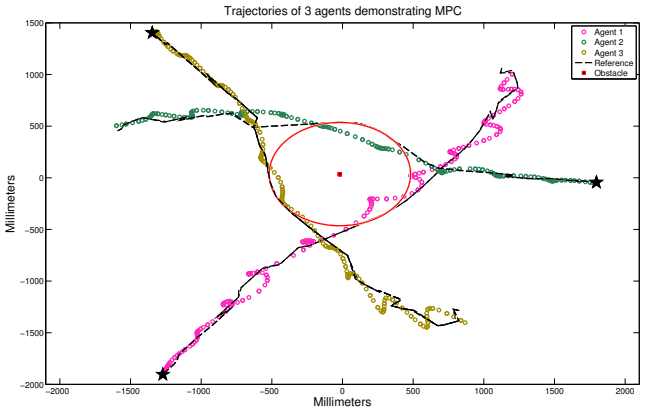

Fig. 7. Path planning reference and actual trajectories for the MPC–SCP demonstration. Note that ⋆ represents the beginning of each trajectory.

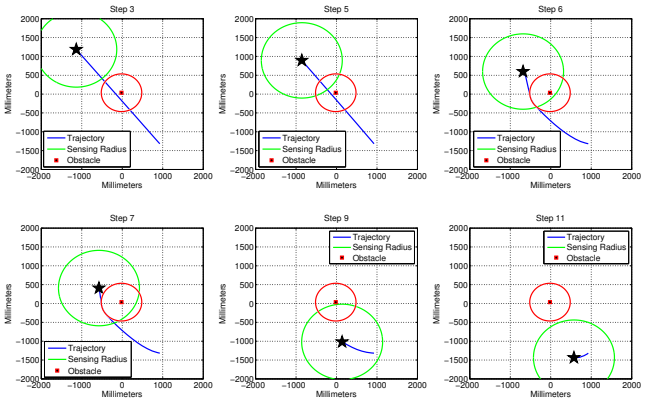

Fig. 8. Real-time reference trajectory, sensing radius, and collision radius for a single agent and obstacle in the MPC–SCP demonstration.

the largest error being less than 200 mm.

The difference between the MPC–SCP trajectories and the SCP trajectories from the previous subsection is shown in Fig. 8. Fig. 8 shows the reference trajectory (blue line) of a single agent (star) at various time steps during the MPC–SCP demonstration. Additionally, the sensing radius (green circle) of the agent and the collision radius (red circle) of the obstacle are shown at each time step. Initially, the agent cannot sense the obstacle so collision avoidance is not considered and the reference trajectory passes through the obstacle. At time step 6, the obstacle is within the sensing radius of the agent so collision avoidance is considered and the reference trajectory avoids the obstacle. The agent continues to update its trajectory, taking into account its actual position, as it approaches the desired terminal position. The limited sensing radius and continuously updating trajectory are what differentiates the MPC–SCP demonstration from the offline path planning done by SCP in the previous subsection.

C. PSG–MPC

The PSG–MPC demonstration uses four agents which are frequently reassigned into four bins. The agents begin at random positions and are initially assigned into one of four bins independently of the other agents. In the following results, each agent has equal probability of going to each bin. The bins are the four quadrants of the $x-y$ plane. As in the path planning scenario, the agents must maintain a distance of 350 mm from one another. The results of this demonstration are shown in Fig. 9–Fig. 10.

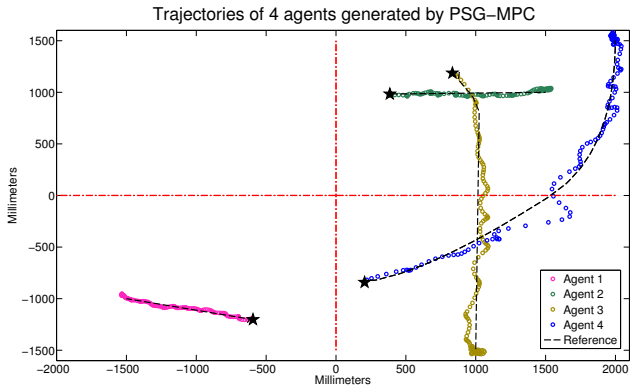

Fig. 9. Path planning reference and actual trajectories for step 1 of the PSG–MPC reassignment. Note that ⋆ represents the beginning of each trajectory.

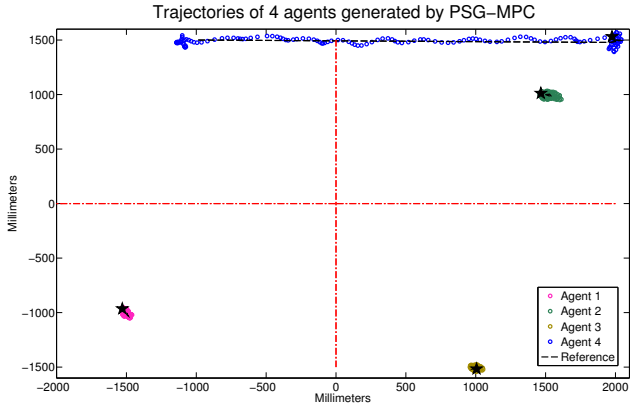

Fig. 10. Path planning reference and actual trajectories for step 2 of the PSG–MPC reassignment. Note that ⋆ represents the beginning of each trajectory.

Fig. 9–Fig. 10 show the reference trajectories (dashed lines) produced by the MPC–SCP algorithm and the actual trajectories (circles) traversed by the agents in relation to the four bins. The four bins are separated by the red (dot-dashed) lines. Results from two iterations of the PSG–MPC algorithm are shown. A time lapse image of the quadrotors at the end of each iteration of PSG–MPC algorithm is shown in Fig. 11. The output shown in Fig. 9–Fig. 10 is the transition of the quadrotors from initialization to step 1 and step 1 to step 2, respectively. The PSG–MPC algorithm converged to the desired formation in 2 iterations. The reference trajectories calculated by MPC–SCP maintain the required distance between agents while taking each agent to its desired bin. Additionally, the agents follow the reference trajectories accurately with the largest error being less than 200 mm.

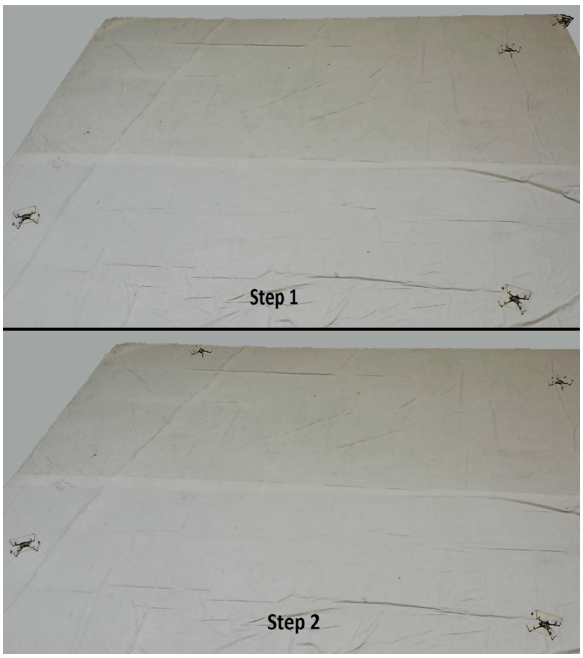

Fig. 11. Quadrotor positions at end of each iteration of the PSG–MPC algorithm.

VI. Conclusion

In this paper, we integrated and implemented the PSG–MPC and MPC–SCP algorithms to reconfigure a swarm of vehicles while avoiding interagent collisions. The PSG–MPC algorithm determined the bin locations for each agent and then called the MPC–SCP algorithm to generate optimal, collision-free trajectories to transfer each agent to its desired bin. This process was repeated until the swarm converged to its desired shape. The integration of the PSG–MPC and MPC–SCP algorithms provided the collision-free trajectories needed to reconfigure the swarm to the desired shape while minimizing the computational and communication burden on the agents.

Additionally, we validated the PSG–MPC and MPC–SCP algorithms in simulation and on our FFT. The simulation results validated the effectiveness of the algorithms on a 200-agent swarm and the experimental results showed that the algorithms can be run in real time with hardware for 4 agents.

References

[1] S. Martínez, J. Cortés, and F. Bullo, "Motion coordination with distributed information," *IEEE Control Syst. Mag.*, vol. 27, no. 4, pp. 75–88, 2008.

[2] G. Arslan, J. R. Marden, and J. S. Shamma, "Autonomous vehicle-target assignment: a game theoretical formulation," *ASME J. Dynamic Systems, Measurement and Control*, vol. 129, pp. 584–596, September 2007.

[3] B. J. Julian, M. Angermann, M. Schwager, and D. Rus, "Distributed robotic sensor networks: An information-theoretic approach," *Inter. J. of Robotics Research*, vol. 31, no. 10, pp. 1134–1154, 2012.

[4] A. Richards, T. Schouwenaars, J. How, and E. Feron, "Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming," *Journal of Guidance, Control, and Dynamics*, vol. 25, no. 4, pp. 755–764, 2002.

[5] F. Y. Hadaegh, S.-J. Chung, and H. M. Manaroha, "On development of 100-gram-class spacecraft for swarm applications," *IEEE Systems Journal (to appear)*, 2014, http://arcl.ae.illinois.edu/FemtoSat_Swarm_Final2013.pdf.

[6] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," *Swarm Robotics: Lecture Notes in Computer Science*, vol. 3342, pp. 10–20.

[7] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: a review from the swarm engineering perspective," *Swarm Intelligence*, vol. 7, no. 1, pp. 1–41, 2013.

[8] J. Seyfried, M. Szymanski, N. Bender, R. Estaña, M. Thiel, and H. Wörn, "The i-swarm project: Intelligent small world autonomous robots for micro-manipulation," *Swarm Robotics: Lecture Notes in Computer Science*, vol. 3342, pp. 70–83, 2005.

[9] P. Kingston and M. Egerstedt, "Index-free multiagent systems: An eulerian approach," in *2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, Annecy, France, 2010, pp. 215–220.

[10] D. Milutinović and P. Lima, "Modeling and optimal centralized control of a large-size robotic population," *IEEE Trans. Robotics*, vol. 22, no. 6, pp. 1280–1285, 2006.

[11] C. C. Cheah, S. P. Hou, and J. J. E. Slotine, "Region-based shape control for a swarm of robots," *Automatica*, vol. 45, no. 10, pp. 2406–2411, 2009.

[12] S. Zhao, S. Ramakrishnan, and M. Kumar, "Density-based control of multiple robots," in *Amer. Control Conf.*, San Francisco, USA, 2011.

[13] M. A. Hsieh, V. Kumar, and L. Chaimowicz, "Decentralized controllers for shape generation with robotic swarms," *Robotica*, vol. 26, pp. 691–701, 2008.

[14] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple robots," *Int. Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, May 2012.

[15] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Probabilistic swarm guidance using inhomogeneous Markov chains," 2014, (in preparation, http://arxiv.org/abs/1403.4134).

[16] S. Berman, A. Halasz, M. A. Hsieh, and V. Kumar, "Optimized stochastic policies for task allocation in swarms of robots," *IEEE Trans. Robotics*, vol. 25, no. 4, pp. 927–937, 2009.

[17] B. Acikmese and D. S. Bayard, "A markov chain approach to probabilistic swarm guidance," in *American Control Conference*, June 2012, pp. 6300–6307.

[18] I. Chattopadhyay and A. Ray, "Supervised self-organization of homogeneous swarms using ergodic projections of markov chains," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 39, no. 6, pp. 1505–1515, 2009.

[19] S. Bandyopadhyay, S.-J. Chung, and F. Y. Hadaegh, "Inhomogeneous markov chain approach to probabilistic swarm guidance algorithm," in *5th Int. Conf. Spacecraft Formation Flying Missions and Technologies*, Munich, Germany, 2013.

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[21] J. Mattingley, Y. Wang, and S. Boyd, "Receding horizon control: Automatic generation of high-speed solvers," *IEEE Control Systems Magazine*, vol. 31, no. 3, pp. 52–65, 2011.

[22] B. Acikmese, D. P. Scharf, E. A. Murray, and F. Y. Hadaegh, "A convex guidance algorithm for formation reconfiguration," in *AIAA Guidance, Navigation, and Control Conference*, Keystone, Colorado, August 2006.

[23] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization," in *Robotics: Science and Systems*, Berlin, Germany, June 2013.

[24] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *International Conference on Intelligent Robots and Systems*, 2012.

[25] R. H. Byrd, J. C. Gilbert, and J. Nocedal, "A trust region method based on interior point techniques for nonlinear programming," *Math. Program. A*, vol. 89, no. 1, pp. 149–185, 2000.

[26] D. Morgan, S.-J. Chung, and F. Y. Hadaegh, "Model predictive control of swarms of spacecraft using sequential convex programming," *Journal of Guidance, Control, and Dynamics*, 2014, (http://arc.aiaa.org/doi/abs/10.2514/1.G000218).

[27] S. Bandyopadhyay and S.-J. Chung, "Distributed estimation using

bayesian consensus filtering," *IEEE Trans. Autom. Control (under review)*, 2014, (http://arxiv.org/abs/1403.3117).
[28] E. Torgerson, *Comparison of Statistical Experiments*. Cambridge University Press, 1991.
[29] P. Billingsley, *Probability and measure*. New York: J. Wiley & Sons, 1995.
[30] M. Valenti, B. Bethke, G. Fiore, H. J. P., and E. Feron, "Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2006.
[31] A. Bürkle, F. Segor, and M. Kollmann, "Towards autonomous micro uav swarms," *Journal of intelligent and robotic systems*, vol. 61, no. 1-4, pp. 339–353, 2011.
[32] A. Kushleyev, D. Mellinger, C. Powers, and V. Kumar, "Towards a swarm of agile micro quadrotors," pp. 287–300, 2012.